Processo Seletivo CROSSBOTS 2022.2 – Candidato 121

Problemas muito simples:

1) A função vai receber os 2 pontos p1 e p2 como strings que contêm 2 coordenadas cada (x e y), e retornar um valor float para a distância utilizando a fórmula. Deixei implícito que p1[0] e p2[0] seriam as coordenadas x e que p1[1] e p2[1] as y.

2) Uma função bem simples, que recebe um valor float para a temperatura em fahrenheits e retorna outro valor float para a temperatura em celsius usando a fórmula.

Problemas simples:

1) Foram chamadas as variáveis referentes à lista, ao valor de x e y e à soma final (junto com um índice i para auxiliar a leitura da lista). Após ler a lista, o valor de x e y, ele realiza a soma dos valores lista[x-1] e lista[y-1] (-1 porque a array começa na posição 0) e os imprime.

2) Inicia a variável somaDiv que se refere a soma dos divisores do número (num) como 0 (ainda não foram encontrados divisores). Soma-se o índice i até que ele chegue no valor logo antes do número, toda vez que o número for divisível por i, acrescentasse i na soma dos divisores. No final, se a soma dos divisores for igual a num, trata-se de um número perfeito, retornando 1 (true), caso contrário retorna 0 (false).

3) Foram criadas variáveis para representar a soma dos divisores (somaDiv), o possível divisor (div), e a lista de números primos encontrados foi recebida como ponteiro a ser alterado (*listaPrimos). Também foi criado o índice i para percorrer a lista original e o índice j para percorrer a lista de primos a ser criada. Para cada número da lista, o valor de div é aumentado de 1 até o valor anterior ao número, se ele for divisor, é adicionado à soma dos divisores daquele número (que volta a ser 0 para cada novo termo da lista). No final desse processo se a soma dos divisores for igual ao próprio número + 1, trata-se de um primo, pois foi divido apenas por ele mesmo e 1, e será adicionado na lista de primos. Retorna a lista de primos encontrados através da recursividade de ponteiros.

4) A função recebe a letra e a frase (string), é criada a variável do contador (iniciada em 0) e um índice auxiliar i. Com ajuda do índice, percorre a frase e quando encontra a letra, tanto maiúscula (toupper) como minúscula (tolower), acrescenta no contador. Quando a string acaba (‘\0’), retorna o valor do contador.

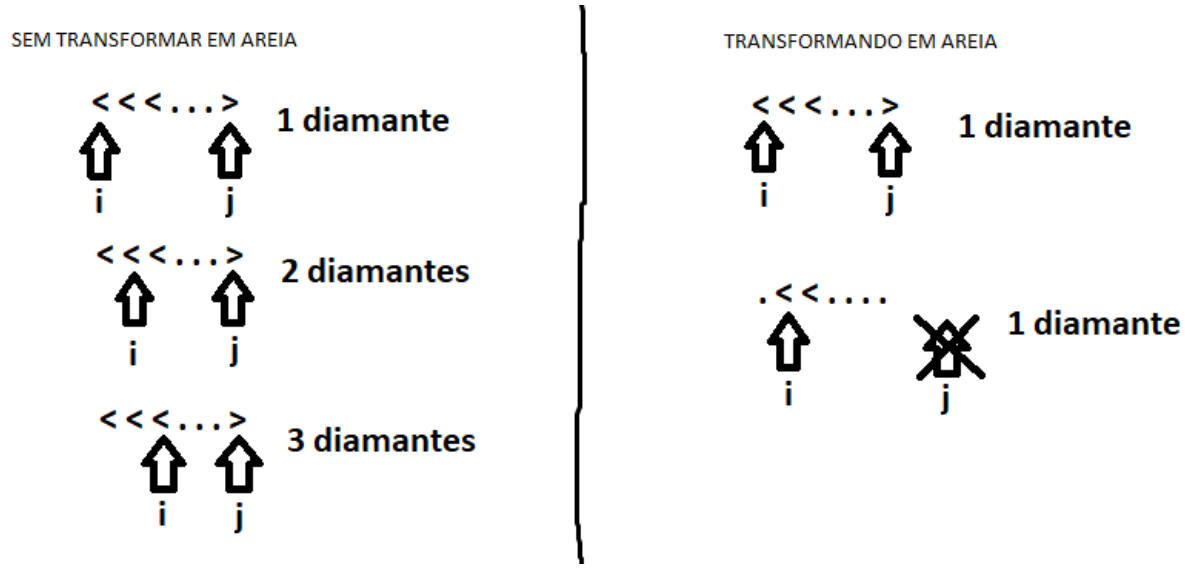
Problemas intermediários:

1) Define o valor de pi antes da função (será usado posteriormente). Percebi que seria mais fácil trabalhar com vetores, já que o produto vetorial á o módulo dos dois vetores vezes o cosseno do angulo entre eles. Criei dois vetores que partem de B e vão para A e C (v e u). Calculei o produto vetorial entre eles e o modulo deles, assim encontrando o cosseno. Usei a função acos que retorna o angulo em radianos, então precisei transformá-lo em graus multiplicando por 180 e dividindo por pi (definido anteriormente).

2) Recebe a lista e a lista onde serão guardados os números que não repetem (*listaNova). Cria a variável rept para ajudar a detectar repetições. Para cada número, o índice j passa por todos os números na sequência, procurando algum igual a ele. Se achar, a variável rept muda seu valor para 1, impossibilitando a transferência desse número para a nova lista. A lista nova é retornada através da recursividade de ponteiros.

Problemas difíceis:

1) Criei dois índices auxiliares para procurar os diamantes e um para saber em qual 'jogo' ele está, junto com o número total de casos de teste (N). Scaneia quantos testes serão feitos e cria um vetor para guardar os resultados de cada teste. A cada jogo cria-se o campo e o scaneia, torna o resultado daquele teste 0, com ajuda do primeiro índice auxiliar (i), passa pelo campo procurando a primeira parte do diamante (<), quando é encontrado usa o segundo índice auxiliar (j) para passar no restante do campo procurando a segunda parte (>) que, se achada, aumenta o número de diamantes achados e transforma as duas partes em areia (.) para não interferir na busca de outros diamantes (como explicado na imagem* abaixo). Imprime os resultados.



*Sim, ela foi feita no paint.

2) Não vou negar, esse problema me deu muita dor de cabeça e me fez questionar se isso valia a pena mesmo. Alguns podem achar que ficou uma gambiarra, eu compreendo, porém aceito isso como uma vitória.

Vamos lá:

Existem dois vetores que interagem entre si para dar o resultado necessário: tabuleiro(char) onde é guardado o tabuleiro lido no console e jogo(int) onde são alterados os 'valores' de cada casa (que começam todas como 0).

O valor de uma casa do tabuleiro é determinado por se alguma peça consegue realizar um movimento e ir até ele. Esse valor é 1 se a peça for preta e 2 se for branca. Não existe a superposição de valores na casa, o valor é dado por qual peça foi lida por último. A leitura é feita de cima para baixo da esquerda para a direita (do início ao fim do vetor tabuleiro). Ex:

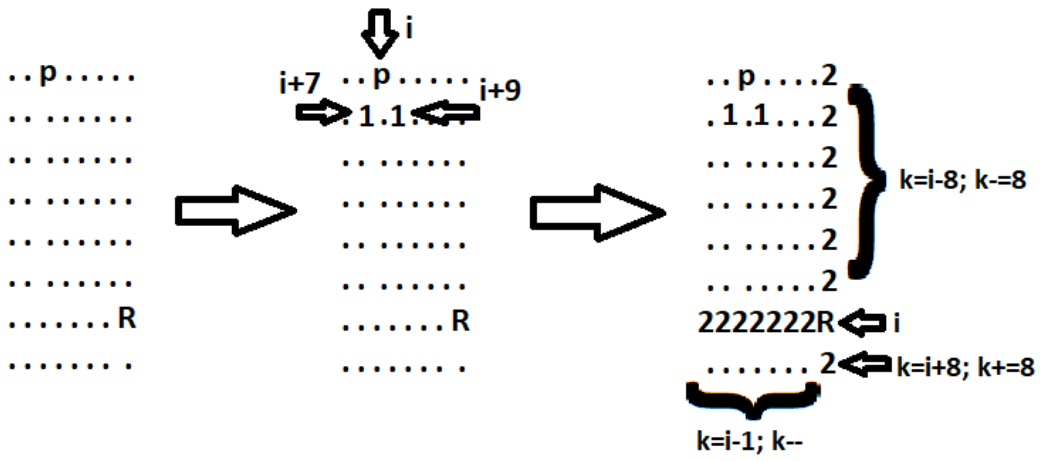
```
...2....
..p2....
.1.2....
...2....
...2....
222R2222
...2....
...2....
```

Por mais que o peão tenha influenciado a casa laranja a ser 1, a torre foi lida depois, tornando ela um 2.

Porém, essa falha não altera o resultado, já que os valores só são alterados se a casa for uma casa vazia ou o rei inimigo, impossibilitando que o rei branco tenha um valor 2 e o rei preto um valor 1 e, também, que ambos estejam em cheque ao mesmo tempo.

Um loop infinito de while(1==1) obriga a leitura e os cálculos a se repetirem até que um tabuleiro inteiro vazio seja lido. Toda vez que um tabuleiro é escrito (8 linhas), é feita uma varredura procurando os reis e definindo suas posições (posreiB – branco, posreiP – preto) e casas vazias, aumentando o número delas (casaVazia++), esse número volta para 0 a cada tabuleiro novo e quando atinge 64 (tabuleiro inteiro vazio) é feito um break para quebrar o loop.

Depois é feita uma segunda varredura, lendo as peças e alterando os valores das casas no vetor jogo de acordo com o movimento das peças. Ex:



índices auxiliares:

i = local da peça sendo lida
k= local das casas a serem modificadas no caso de repetições (torres, bispos e rainhas)

Depois de ler todas as peças e fazer as devidas alterações nos valores das casas, é checado o valor da casa de ambos os reis (posreiB e posreiP), caso o valor da casa do rei branco seja 1, ele está em cheque, fazendo o resultado desse jogo ser 2, caso o valor da casa do rei preto seja 2, ele está em cheque e o resultado desse jogo será 1. Caso nenhuma das duas condições seja verdadeira, o resultado será 0.

Por fim é feita a impressão dos resultados.